

# An Explicit Nonlinear Mapping for Manifold Learning

Hong Qiao, *Senior Member, IEEE*, Peng Zhang, Di Wang, and Bo Zhang

**Abstract**—Manifold learning is a hot research topic in the field of computer science and has many applications in the real world. A main drawback of manifold learning methods is, however, that there is no explicit mappings from the input data manifold to the output embedding. This prohibits the application of manifold learning methods in many practical problems such as classification and target detection. Previously, in order to provide explicit mappings for manifold learning methods, many methods have been proposed to get an approximate explicit representation mapping with the assumption that there exists a linear projection between the high-dimensional data samples and their low-dimensional embedding. However, this linearity assumption may be too restrictive. In this paper, an explicit nonlinear mapping is proposed for manifold learning, based on the assumption that there exists a polynomial mapping between the high-dimensional data samples and their low-dimensional representations. As far as we know, this is the first time that an explicit nonlinear mapping for manifold learning is given. In particular, we apply this to the method of Locally Linear Embedding (LLE) and derive an explicit nonlinear manifold learning algorithm, named Neighborhood Preserving Polynomial Embedding (NPPE). Experimental results on both synthetic and real-world data show that the proposed mapping is much more effective in preserving the local neighborhood information and the nonlinear geometry of the high-dimensional data samples than previous work.

**Index Terms**—Manifold learning, nonlinear dimensionality reduction, machine learning, data mining.

## 1 INTRODUCTION

MANIFOLD learning has drawn great interests since it was first proposed in 2000 ([1], [2], [4]) as a promising nonlinear dimensionality reduction (NDR) method for high-dimensional data manifolds. Its basic assumption is that high-dimensional input data samples lie on or close to a low-dimensional smooth manifold embedded in the ambient Euclidean space. For example, by rotating the camera around the same object with fixed radius, images of the object can be viewed as a one-dimensional curve embedded in a high-dimensional Euclidean space, whose dimension equals to the number of pixels in the image. With the manifold assumption, manifold learning methods aim to extract the intrinsic degrees of freedom underlying the input high-dimensional data samples, by preserving local or global geometric characteristics of the manifold from which

data samples are drawn. In recent years, various manifold learning algorithms have been proposed, such as locally linear embedding (LLE) [2], [3], ISOMAP [4], [5], Laplacian eigenmap (LE) [12], diffusion maps (DM) [14], local tangent space alignment (LTSA) [11], and Riemannian manifold learning [13]. They have achieved great success in finding meaningful low-dimensional embeddings for high-dimensional data manifolds. Meanwhile, manifold learning also has many important applications in real-world problems, such as human motion detection [17], human face recognition [18], classification and compressed expression of hyper-spectral imageries [19], dynamic shape and appearance classification [20], and visual tracking [21]–[23].

However, a main drawback of the manifold learning methods is that they learn the low-dimensional representations of the high-dimensional input data samples implicitly. No explicit mapping relationship from the input data manifold to the output embedding can be obtained after the training process. Therefore, in order to obtain the low-dimensional representations of the new coming samples, the learning procedure, containing all previous samples and new samples as inputs, has to be repeatedly implemented. It is obvious that such a strategy is extremely time-consuming for sequentially arrived data, which greatly limits the application of the manifold learning methods to many practical problems, such as classification, target detection, visual tracking and detection.

In order to address the issue of lacking explicit mappings, many linear projection based methods have been proposed for manifold learning by assuming that there exists a linear projection between the high-dimensional

- H. Qiao is with the Lab of Complex Systems and Intelligent Science, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.  
E-mail: hong.qiao@ia.ac.cn
- P. Zhang and D. Wang are with the Graduate School and the Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing, 100190, China.  
E-mail: {zhangpeng, wangdi}@amss.ac.cn
- B. Zhang is with the State Key Lab of Scientific and Engineering Computing and the Institute of Applied Mathematics, AMSS, Chinese Academy of Sciences, Beijing 100190, China.  
E-Mail: b.zhang@amt.ac.cn

The work of H. Qiao was supported in part by the National Natural Science Foundation (NNSF) of China under Grants 60675039 and 60621001 and by the Outstanding Youth Fund of the NNSF of China under Grant 60725310. The work of B. Zhang was supported in part by the 863 Program of China under Grant 2007AA04Z228, by the 973 Program of China under Grant 2007CB311002 and by the NNSF of China under Grant 90820007.

input data samples and their low-dimensional representations, such as Locality Preserving Projections (LPP) [24], [25], Neighborhood Preserving Embedding (NPE) [26], Neighborhood Preserving Projections (NPP) [27], Orthogonal Locality Preserving Projections (OLPP) [28], Orthogonal Neighborhood Preserving Projections (ONPP) [29], [30], and Graph Embedding [31]. Although these methods have achieved their success in many problems, the linearity assumption may still be too restrictive.

On the other hand, several kernel-based methods have also been proposed to give nonlinear but implicit mappings for manifold learning (see, e.g. [32]–[35]). These methods reformulate the manifold learning methods as kernel learning problems and then utilize the existing kernel extrapolation techniques to find the location of new data samples in the low-dimensional space. The mappings provided by the kernel-based methods are nonlinear and implicit. Furthermore, the performance of these methods depends on the choice of the kernel functions, and their computational complexity is extremely high for very large data sets.

In this paper, an explicit nonlinear mapping for manifold learning is proposed for the first time, based on the assumption that there exists a polynomial mapping from the high-dimensional input data samples to their low-dimensional representations. The proposed mapping has the following main features.

- 1) The mapping is explicit, so it is straightforward to locate any new data samples in the low-dimensional space. This is different from the traditional manifold learning methods such as like LLE, LE, and ISOMAP [4] in which the mapping is implicit and it is not clear how new data samples can be embedded in the low-dimensional space. Compared with kernel-based mappings, the proposed mapping does not depend on the specific kernels in finding the low-dimensional representations of new data samples.
- 2) The mapping is nonlinear. In contrast to the linear projection-based methods which find a linear projection mapping from the input high-dimensional samples to their low-dimensional representations, the proposed mapping provides a nonlinear polynomial mapping from the input space to the reduced space. Clearly, it is more reasonable to use a polynomial mapping to handle with data samples lying on nonlinear manifolds. Meanwhile, our analysis and experiments show that the proposed mapping is of similar computational complexity with the linear projection-based methods.

Combining this explicit nonlinear mapping with existing manifold learning methods (e.g. LLE, LE, Isomap) can give explicit manifold learning algorithms. In this paper, we concentrate on the LLE manifold learning method and propose an explicit nonlinear manifold learning algorithm called Neighborhood Preserving Polynomial Embedding (NPPE) algorithm. Experiments

TABLE 1  
Main notations

$\mathbb{R}^n$	$n$ -dimensional Euclidean space where input samples lie
$\mathbb{R}^m$	$m$ -dimensional Euclidean space, $m < n$ , where the low-dimensional embedding lie
$x_i$	$x_i = (x_i^1, \dots, x_i^n)^T$ , the $i$ -th input sample in $\mathbb{R}^n$ , $i = 1, 2, \dots, N$
$\mathcal{X}$	$\mathcal{X} = \{x_1, x_2, \dots, x_N\}$ , the set of input samples
$X$	$X = [x_1 \ x_2 \ \dots \ x_N]$ , $n \times N$ matrix of input samples
$y_i$	$y_i = (y_i^1, \dots, y_i^m)^T$ , low-dimensional representation of $x_i$ obtained by manifold learning, $i = 1, 2, \dots, N$
$\mathcal{Y}$	$\mathcal{Y} = \{y_1, y_2, \dots, y_N\}$ , the set of low-dimensional representations
$Y$	$Y = [y_1 \ y_2 \ \dots \ y_N]$ , $m \times N$ matrix of low-dimensional representations
$I_m$	Identity matrix of size $m$
$\ \cdot\ _2$	$L_2$ -norm where $\ v\ _2 = \sqrt{\sum_{k=1}^m (v^k)^2}$ for an $m$ -dimensional vector $v$

on both synthetic and real-world data have been conducted to illustrate the validity and effectiveness of the proposed mapping.

The remaining part of the paper is organized as follows. Section 2 gives a brief review of the existing manifold learning methods including those based on linear projections and kernel-based nonlinear mappings. Details of the explicit nonlinear mapping for manifold learning are presented in Section 3, whilst the NPPE algorithm is given in Section 4. In Section 5, experiments are conducted on both synthetic and real-world data sets to demonstrate the validity of the proposed algorithm. Conclusion is given in Section 6.

## 2 RELATED WORKS

In this section, we briefly review existing manifold learning algorithms including those based on linear projections and out-of-sample nonlinear extensions for learned manifolds.

For convenience of presentation, the main notations used in this paper are summarized in Table 1. Throughout this paper, all data samples are in the form of column vectors. Matrices are expressed using normal capital letters and data vectors are represented using lowercase letters. The superscript of a data vector is the index of its component.

### 2.1 Manifold Learning Methods

According to the geometric characteristics which are preserved, existing manifold learning methods can be cast into two categories: local or global approaches.

As local approaches, Locally Linear Embedding (LLE) [2], [3] preserves local reconstruction weights. Locally Multidimensional Scaling (LMDS) [9] preserves local pairwise Euclidean distances among data samples. Maximum Variance Unfolding (MVU) [10] also preserves pairwise Euclidean distances in each local neighborhood, but it maximizes the variance of the low-dimensional representations at the same time. Local Tangent Space

Alignment (LTSA) [11] keeps the local tangent structure. Diffusion Maps [14] preserves local pairwise diffusion distances from high-dimensional data to the low-dimensional representations. Laplacian Eigenmap (LE) [12] preserves the local adjacency relationship.

As global approaches, Isometric Feature Mapping (ISOMAP) [4], [5] preserves the pairwise geodesic distances among the high-dimensional data samples and their low-dimensional representations. Hessian Eigenmaps (HLE) [15] extends ISOMAP to more general cases where the set of intrinsic degrees of freedom may be non-convex. In Riemannian Manifold Learning (RML) [13], the coordinates of data samples in the tangential space are preserved to be their low-dimensional representations.

## 2.2 Linear Projections for Manifold Learning

Manifold learning algorithms based on linear projections assume that there exists a linear projection which maps the high-dimensional samples into a low-dimensional space, that is,

$$y_i = U^T x_i, \text{ where } U \in \mathbb{R}^{n \times m}, \quad (1)$$

where  $x_i$  is a high-dimensional sample and  $y_i$  is its low-dimensional representation. Denote by  $u_i$  the  $i$ -th column of  $U$ . Then from a geometric point of view, data samples in  $\mathbb{R}^n$  are projected into an  $m$ -dimensional linear subspace spanned by  $\{u_i\}_{i=1}^n$ . The low-dimensional representation  $y_i$  is the coordinate of  $x_i$  in  $\mathbb{R}^m$  with respect to the basis  $\{u_i\}_{i=1}^n$ .

### 2.2.1 LPP

Locality Preserving Projections (LPP) [24], [25] provides a linear mapping for Laplacian Eigenmaps (LE), by applying (1) into the training procedure of LE. The LE method aims to train a set of low-dimensional representations  $\mathcal{Y}$  which can best preserve the adjacency relationship among high-dimensional inputs  $\mathcal{X}$ . If  $x_i$  and  $x_j$  are "close" to each other, then  $y_i$  and  $y_j$  should also be so. This property is achieved by solving the following constrained optimization problem

$$\min \sum_{i,j=1}^N W_{ij} \|y_i - y_j\|_{L_2}^2 \quad (2)$$

$$\text{s. t. } \sum_{i=1}^N D_i y_i y_i^T = I_m, \quad (3)$$

where the penalty weights  $W_{ij}$  are given by the heat kernel  $W_{ij} = \exp(-\|x_i - x_j\|_2^2/t)$  and  $D_i = \sum_j W_{ij}$ .

In LPP, equation (1) is applied to (2) and (3), that is, each  $x_i$  is replaced with  $U^T y_i$ . By a straightforward algebraic calculation, equations (2) and (3) are transformed into

$$\min \text{Tr}(U^T X L X^T U) \quad (4)$$

$$\text{s. t. } U^T X D X^T U = I_m, \quad (5)$$

where  $W = (W_{ij})$ ,  $L = D - W$  and  $D$  is the diagonal matrix whose  $(i, i)$ -th entry is  $D_i$ . This optimization problem leads to a generalized eigenvalue problem

$$X L X^T u_i = \lambda_i X D X^T u_i,$$

and the optimal solutions  $u_1, u_2, \dots, u_m$  are the eigenvectors corresponding to the  $m$  smallest eigenvalues.

Once  $\{u_i\}_{i=1}^n$  are computed, the linear projection matrix provided by LPP is given by  $U = [u_1 \ u_2 \ \dots \ u_m]$ . For any new data sample  $x$  from the high-dimensional space  $\mathbb{R}^n$ , LPP finds its low-dimensional representation  $y$  simply by  $y = U^T x$ .

### 2.2.2 NPP and NPE

The linear projection mapping for Locally Linear Embedding (LLE) is independently provided by Neighborhood Preserving Embedding (NPE) [26] and Neighborhood Preserving Projections (NPP) [27]. Similarly to LPP, NPE and NPP apply the linear projection assumption (1) to the training process of LLE and reformulate the optimization problem in LLE as to compute the linear projection matrix.

During the training procedure of LLE, a set of linear reconstruction weights  $\{W_{ij}\}_{i,j=1}^N$  are first computed by solving a convex optimization problem

$$\begin{aligned} \min \quad & \sum_{i=1}^N \|x_i - \sum_{j=1}^N W_{ij} x_j\|_2^2 \\ \text{s. t. } \quad & W_{ij} = 0, \text{ if } j \notin N(i) \\ & \sum_{j=1}^N W_{ij} = 1, \end{aligned}$$

where  $N(i)$  is the index set of the  $k$  nearest neighbors of  $x_i$ . Then LLE aims to preserve  $\{W_{ij}\}_{i,j=1}^N$  from  $\mathcal{X}$  to  $\mathcal{Y}$ . This is achieved by solving the following optimization problem

$$\min \sum_{i=1}^N \|y_i - \sum_{j=1}^N W_{ij} y_j\|_2^2 \quad (6)$$

$$\text{s. t. } \frac{1}{N} \sum_{i=1}^N y_i y_i^T = I_m \quad (7)$$

In NPE and NPP, the linear projection assumption (1) is used in the above optimization problem, so (6) and (7) become

$$\min \text{Tr}(U^T X M X^T U) \quad (8)$$

$$\text{s. t. } U^T X X^T U = I_m \quad (9)$$

where  $M = (I_N - W)^T (I_N - W)$  with  $W = (W_{ij})$ . The optimal solutions  $u_1, u_2, \dots, u_m$  are the eigenvectors of the following generalized eigenvalue problem corresponding to the  $m$  smallest eigenvalues

$$X M X^T u_i = \lambda_i X X^T u_i.$$

After finding the linear projection matrix  $U = [u_1 \ u_2 \ \cdots \ u_m]$ , any new data sample  $x$  from the high-dimensional space  $\mathbb{R}^n$  can be easily mapped into the lower dimensional space  $\mathbb{R}^m$  by  $y = U^T x$ .

### 2.2.3 OLPP and ONPP

Orthogonal Locality Preserving Projections (OLPP) [28] and Orthogonal Neighborhood Preserving Projections (ONPP) [29], [30] are the same as LPP and NPE (or NPP), respectively, except that the linear projection matrix provided by LPP and NPE (or NPP) is restricted to be orthogonal. This is achieved by replacing the constraints (5) and (9) with  $U^T U = I_m$ . Then the optimization problems in OLPP and ONPP become

$$\text{OLPP: } U_{OLPP} = \underset{U^T U = I_m}{\operatorname{argmin}} \operatorname{Tr}(U^T X L X^T U) \quad (10)$$

$$\text{ONPP: } U_{ONPP} = \underset{U^T U = I_m}{\operatorname{argmin}} \operatorname{Tr}(U^T X M X^T U) \quad (11)$$

Unlike in the cases of LPP and NPE (or NPP), these two optimization problems lead to eigenvalue problems which are much easier to solve numerically than a generalized eigenvalue problem. The column vectors of  $U_{OLPP}$  are given by the eigenvectors of  $X L X^T$  corresponding to the  $m$  smallest eigenvalues. The same result holds for  $U_{ONPP}$  by replacing  $X L X^T$  with  $X M X^T$ . The reader is referred to [28] and [29], [30] for details of these two algorithms.

## 2.3 Out-of-Sample Nonlinear Extensions for Manifold Learning

Besides linear projections for manifold learning, several out-of-sample nonlinear extensions are also proposed for manifold learning in order to get low-dimensional representations of unseen data samples from the learned manifold. These methods are based on kernel functions and extrapolation techniques. A common strategy taken by these methods is to reformulate manifold learning methods as kernel learning problems. Then extrapolation techniques are employed to find the location of new coming samples in the low-dimensional space from the learned manifold. Bengio et al. [32], [36] proposed a unified framework for extending LLE, ISOMAP and LE, in which these methods are seen as learning eigenfunctions of operators defined from data-dependent kernels. The data-dependent kernels are implicitly defined by LLE, ISOMAP LE and are used together with the Nyström formula [38] to extrapolate the embedding of a manifold learned from finite training samples to new coming samples for LLE, ISOMAP and LE (see [32], [36]). Chin and Suter [35] investigated the equivalence between MVU and Kernel Principal Component Analysis (KPCA) [39], by which extending MVU to new samples is reduced to extending a kernel matrix. In their work [35], the kernel matrix is generated from an unknown kernel eigenfunction which is approximated using Gaussian basis functions. A framework was proposed in [33]

for efficient kernel extrapolation which is based on a matrix approximation theorem and an extension of the representer theorem. Under this framework, LLE was reformulated and the issue of extending LLE to new data samples was addressed in [33].

## 3 EXPLICIT NONLINEAR MAPPINGS FOR MANIFOLD LEARNING

In this section, we propose an explicit nonlinear mapping for manifold learning, based on the assumption that there is a polynomial mapping between the high-dimensional data samples and their lower dimensional representations. Precisely, given input samples  $x_1, x_2, \dots, x_N$  and their low dimensional representations  $y_1, y_2, \dots, y_N$ , we assume that there exists a polynomial mapping which maps  $\mathcal{X}$  to  $\mathcal{Y}$ , that is, the  $k$ -th component  $y_i^k$  of  $y_i$  is a polynomial of degree  $p$  with respect to  $x_i$  in the following manner:

$$y_i^k = \sum_{\substack{l_1, l_2, \dots, l_n \geq 0 \\ 1 \leq l_1 + l_2 + \dots + l_n \leq p}} v_k^1(x_i^1)^{l_1} (x_i^2)^{l_2} \cdots (x_i^n)^{l_n}, \quad (12)$$

where  $l_1, l_2, \dots, l_n$  are all integers. The superscript 1 stands for the  $n$ -tuple indexing array  $(l_1, l_2, \dots, l_n)$  and  $v_k$  is the vector of polynomial coefficients which is defined by

$$v_k = \begin{pmatrix} v_k^1 |_{l_1=p, l_2=0, \dots, l_n=0} \\ v_k^1 |_{l_1=p-1, l_2=1, \dots, l_n=0} \\ \vdots \\ v_k^1 |_{l_1=1, l_2=0, \dots, l_n=0} \\ \vdots \\ v_k^1 |_{l_1=0, l_2=0, \dots, l_n=1} \end{pmatrix}. \quad (13)$$

By assuming the polynomial mapping relationship, we aim to find a polynomial approximation to the unknown mapping from the high-dimensional data samples into their low-dimensional embedding space. Compared with the linear projection assumption used previously, a polynomial mapping provides high-order approximation to the unknown nonlinear mapping and therefore is more accurate for data samples lying on nonlinear manifolds.

In order to apply this explicit nonlinear mapping to manifold learning algorithms, we need two definitions from matrix analysis [40].

*Definition 3.1:* The Kronecker product of an  $m \times n$  matrix  $A$  and a  $p \times q$  matrix  $B$  is defined as

$$A \otimes B = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

which is an  $mp \times nq$  matrix.

*Definition 3.2:* The Hadamard product of two  $m \times n$  matrices  $A$  and  $B$  is defined as

$$A \odot B = \begin{pmatrix} a_{11}b_{11} & \cdots & a_{1n}b_{1n} \\ \vdots & & \vdots \\ a_{m1}b_{m1} & \cdots & a_{mn}b_{mn} \end{pmatrix}$$



Recently, it was proved in [31] that most manifold learning methods, including LLE, LE, and ISOMAP, can be cast into the framework of spectral embedding. Under this framework, finding the low-dimensional embedding representations of the high-dimensional data samples is reduced to solving the following optimization problem

$$\min_{y_i} \frac{1}{2} \sum_{i,j=1}^N W_{ij} \|y_i - y_j\|_2^2 \quad (14)$$

$$\text{s. t.} \quad \sum_{i=1}^N D_i y_i y_i^T = I_m \quad (15)$$

where  $W_{ij}$ ,  $i, j = 1, 2, \dots, N$ , are positive weights which can be defined by using the input data samples and  $D_i = \sum_{j=1}^N W_{ij}$ .

Applying the polynomial assumption (12) to the above general model of manifold learning gives a general manifold learning algorithm with an explicit nonlinear mapping. Denote  $(x_i^1)^{l_1} (x_i^2)^{l_2} \dots (x_i^n)^{l_n}$  by  $x_i^1$  and substitute (12) into (14). Then the objective function becomes

$$\begin{aligned} & \frac{1}{2} \sum_{i,j} W_{ij} \left\| \begin{pmatrix} \sum_1 v_1^1 x_i^1 \\ \vdots \\ \sum_1 v_m^1 x_i^1 \end{pmatrix} - \begin{pmatrix} \sum_1 v_1^1 x_j^1 \\ \vdots \\ \sum_1 v_m^1 x_j^1 \end{pmatrix} \right\|_2^2 \\ &= \frac{1}{2} \sum_{i,j} W_{ij} \sum_k \left( \left( \sum_1 v_k^1 x_i^1 \right) - \left( \sum_1 v_k^1 x_j^1 \right) \right)^2 \\ &= \sum_{i,j} W_{ij} \sum_k \left( \left( \sum_1 v_k^1 x_i^1 \right)^2 - \left( \sum_1 v_k^1 x_i^1 \right) \left( \sum_1 v_k^1 x_j^1 \right) \right) \\ &= \sum_k \left( \sum_i \left( \sum_1 v_k^1 x_i^1 \right) \left( \sum_j W_{ij} \right) \left( \sum_1 v_k^1 x_i^1 \right) \right) \\ &\quad - \sum_k \left( \sum_{i,j} \left( \sum_1 v_k^1 x_i^1 \right) W_{ij} \left( \sum_1 v_k^1 x_j^1 \right) \right) \\ &= \sum_k \left( \sum_i \left( \sum_1 v_k^1 x_i^1 \right) D_i \left( \sum_1 v_k^1 x_i^1 \right) \right) \\ &\quad - \sum_k \left( \sum_{i,j} \left( \sum_1 v_k^1 x_i^1 \right) W_{ij} \left( \sum_1 v_k^1 x_j^1 \right) \right) \quad (16) \end{aligned}$$

Substitute (12) into (15), so the constraint is transformed into

$$\sum_i D_i \begin{pmatrix} \sum_1 v_1^1 x_i^1 \\ \vdots \\ \sum_1 v_m^1 x_i^1 \end{pmatrix} \begin{pmatrix} \sum_1 v_1^1 x_i^1 \cdots \sum_1 v_m^1 x_i^1 \end{pmatrix} = I_m$$

This is equivalent to

$$\sum_i D_i \left( \sum_1 v_j^1 x_i^1 \right) \left( \sum_1 v_k^1 x_i^1 \right) = \delta_{jk} \quad (17)$$

where  $\delta_{jk} = 1$  for  $j = k$  and  $= 0$  otherwise.

In order to simplify (16) and (17), we define  $X_p^{(i)}$  by

$$X_p^{(i)} = \begin{pmatrix} \overbrace{x_i \otimes x_i \otimes \dots \otimes x_i}^p \\ \vdots \\ x_i \otimes x_i \\ x_i \end{pmatrix}. \quad (18)$$

Then  $\sum_1 v_k^1 x_i^1 = v_k^T X_p^{(i)}$ , so (16) and (17) are reduced, respectively, to

$$\min_{v_k} \sum_k v_k^T \left\{ \sum_i X_p^{(i)} D_i (X_p^{(i)})^T - \sum_{ij} X_p^{(i)} W_{ij} (X_p^{(j)})^T \right\} v_k \quad (19)$$

$$\text{s. t.} \quad v_j^T \left\{ \sum_i X_p^{(i)} D_i (X_p^{(i)})^T \right\} v_k = \delta_{jk} \quad (20)$$

By writing  $X_p = [X_p^{(1)} X_p^{(2)} \dots X_p^{(N)}]$ , (19) and (20) can be further simplified to

$$\min_{v_k} \sum_k v_k^T X_p W X_p v_k \quad (21)$$

$$\text{s. t.} \quad v_j^T X_p D X_p v_k = \delta_{jk}, \quad (22)$$

where  $W = (W_{ij})$  and  $D$  is a diagonal matrix whose  $i$ -th diagonal entry is  $D_i$ .

By the Rayleigh-Ritz Theorem [40], the optimal solutions  $v_k$ ,  $k = 1, 2, \dots, m$ , are the eigenvectors of the following generalized eigenvalue problem corresponding to the  $m$  smallest eigenvalues

$$X_p (D - W) X_p^T v_i = \lambda X_p D X_p^T v_i, \quad v_i^T X_p D X_p^T v_j = \delta_{ij} \quad (23)$$

Once  $v_k$ ,  $k = 1, 2, \dots, m$ , are computed, the explicit nonlinear mapping from the high-dimensional data samples to the low-dimensional embedding space  $\mathbb{R}^m$  can be given as

$$y = \begin{pmatrix} \sum_1 v_1^1 (x^1)^{l_1} (x^2)^{l_2} \dots (x^n)^{l_n} \\ \vdots \\ \sum_1 v_m^1 (x^1)^{l_1} (x^2)^{l_2} \dots (x^n)^{l_n} \end{pmatrix}, \quad (24)$$

where  $x$  is a high-dimensional data sample and  $y$  is its low-dimensional representation. For a new coming sample  $x_{new}$ , its location in the low-dimensional embedding manifold can be simply obtained by

$$y_{new} = (v_1^T X_p^{(new)}, v_2^T X_p^{(new)}, \dots, v_m^T X_p^{(new)})^T, \quad (25)$$

where  $X_p^{(new)}$  is defined in the same way as in (18).

In the next section, we will make use of a similar method as in LLE to define the weights  $W_{ij}$ ,  $i, j = 1, 2, \dots, N$ , so that the geometry of the neighborhood of each data point can be captured.

## 4 NEIGHBORHOOD PRESERVING POLYNOMIAL EMBEDDING

In this section, we propose a new manifold learning algorithm with an explicit nonlinear mapping, named Neighborhood Preserving Polynomial Embedding (NPPE), which is obtained by defining the weights  $W_{ij}$ ,  $i, j = 1, 2, \dots, N$ , in a way similar to the LLE method and combining them with the explicit nonlinear mapping as in the preceding Section 3.

### 4.1 NPPE

Consider a data set  $\{x_1, x_2, \dots, x_N\}$  from the high-dimensional space  $\mathbb{R}^n$ . NPPE starts with finding a set of linear reconstruction weights which can best reconstruct each data point  $x_i$  by its  $k$ -nearest neighbors ( $k$ -NNs). This step is identical with that of LLE [2], [3]. The weights  $R_{ij}$ ,  $i, j = 1, 2, \dots, N$ , which are defined to be nonzero only if  $x_j$  is among the  $k$ -NNs of  $x_i$ , are computed by solving the following optimization problem

$$R_{ij} = \underset{\sum_{j=1}^N R_{ij}=1}{\operatorname{argmin}} \sum_{i=1}^N \|x_i - \sum_{j=1}^N R_{ij}x_j\|_2^2. \quad (26)$$

The weights  $R_{ij}$  represent the linear coefficients for reconstructing the sample  $x_i$  from its neighbors  $\{x_j\}$ , whilst the constraint  $\sum_{j=1}^N R_{ij} = 1$  means that  $x_i$  is approximated by a convex combination of its neighbors. The weight matrix,  $R = (R_{ij})$ , has a closed-form solution given by

$$r_i = \frac{G^{-1}e}{e^T G^{-1}e}, \quad (27)$$

where  $r_i$  is a column vector formed by the  $k$  non-zero entries in the  $i$ -th row of  $R$  and  $e$  is a column vector of all ones. The  $(j, l)$ -th entry of the  $k \times k$  matrix  $G$  is  $(x_j - x_i)^T(x_l - x_i)$ , where  $x_j$  and  $x_l$  are among the  $k$ -NNs of  $x_i$ .

NPPE aims to preserve the reconstruction weights  $R_{ij}$  from the high-dimensional input data samples to their low-dimensional representations under the polynomial mapping assumption. This is achieved by solving the following optimization problem

$$\mathcal{Y} = \underset{\sum_{i=1}^N y_i y_i^T = I_m}{\operatorname{argmin}} \sum_{i=1}^N \|y_i - \sum_{j=1}^N R_{ij}y_j\|_2^2, \quad (28)$$

where each  $y_i$  satisfies (12).

By a simple algebraic calculation, it can be shown that (28) is equivalent to (14) and (15) with

$$W_{ij} = R_{ij} + R_{ji} - \sum_{k=1}^N R_{ik}R_{kj}, \text{ and } D_i = 1. \quad (29)$$

By the result in Section 3, the explicit nonlinear mapping can be obtained by solving (23) and the low-dimensional representations  $\mathcal{Y}$  of  $\mathcal{X}$  can be computed by applying (24) to  $\mathcal{X}$ . For a new coming sample  $x_{new}$ , its low-dimensional representation can be simply given by (25).

We conclude this section by summarizing the NPPE algorithm in Algorithm 1.

---

#### Algorithm 1: The NPPE Algorithm

---

**Input:** Data matrix  $X$ , the number  $k$  of nearest neighbors and the polynomial degree  $p$ .

**Output:** Vectors of polynomial coefficients  $v_i$ ,  $i = 1, 2, \dots, m$ .

Compute  $R_{ij}$  by (27).

Compute  $W$  and  $D$  by (29).

Generate  $X_p$  according to (18).

Solve the generalized eigenvalue problem (23) to get  $v_i$ ,  $i = 1, 2, \dots, m$ .

---



---

#### Algorithm 2: The Simplified NPPE Algorithm

---

**Input:** Data matrix  $X$ , the number  $k$  of nearest neighbors and the polynomial degree  $p$ .

**Output:** Vectors of polynomial coefficients  $v_i$ ,  $i = 1, 2, \dots, m$ .

Compute  $R_{ij}$  by (27).

Compute  $W$  and  $D$  by (29).

Generate  $X_p$  according to (30).

Solve the generalized eigenvalue problem (23) to get  $v_i$ ,  $i = 1, 2, \dots, m$ .

---

### 4.2 Computational Complexity and Simplified NPPE

In the training procedure of NPPE, the computational complexity of generating  $X_p$  is  $O(N \sum_{i=2}^p n^i)$ . Computing  $X_p W X_p^T$  and  $X_p D X_p^T$  takes  $O(kN^2 \sum_{i=1}^p n^i)$  and  $O(N^2 \sum_{i=1}^p n^i)$  operations, respectively, since there are only  $k$  non-zero entries in each column of  $W$  and  $D$  is a diagonal matrix. The computational complexity of the final eigen-decomposition is  $O(m(\sum_{i=1}^p n^i)^3)$ , which is the most time-consuming step.

In the procedure of locating new samples with NPPE, generating  $X_p^{(new)}$  takes  $O(\sum_{i=2}^p n^i)$  operations and computing  $y_{new}$  takes  $O(m(\sum_{i=1}^p n^i)^2)$  operations.

From the above analysis, it can be seen that, as the polynomial order  $p$  increases, the overall computational complexity increases exponentially with  $p$ , which would be extremely time-consuming when the data dimension is very high. To address this issue, we simplify NPPE by removing the crosswise items. This is achieved by replacing the Kronecker product in (18) with the Hadamard product

$$X_p^{(i)} = \begin{pmatrix} \overbrace{x_i \odot x_i \odot \dots \odot x_i}^p \\ \vdots \\ x_i \odot x_i \\ x_i \end{pmatrix}. \quad (30)$$

With this strategy, the computational complexity of generating  $X_p$  is reduced to  $O(np(p+1)/2)$ , whilst the computational complexity computing  $y_{new}$  is reduced to  $O(mn^2p^2)$ . The Simplified NPPE (SNPPE) is summarized in Algorithm 2.

Finally, the computational complexity of SNPPE, linear methods and kernel methods on computing  $y_{new}$  is

TABLE 2

Computational complexity of SNPPE, linear methods and kernel methods on computing the low-dimensional representation of a new coming sample.

Methods	SNPPE	Linear	Kernel
Complexity	$O(mn^2p^2)$	$O(mn^2)$	$O(n^2N^2)$

summarized in Table 2. The computational complexity of different kernel methods varies. Here we only state the computational complexity of the common step of computing the inner products. It is obvious that the total complexity in computing  $y_{new}$  is not less than this value.

### 4.3 Discussion

In this subsection, we briefly explain why NPPE or SNPPE has a better performance than its linear counterparts for nonlinearly distributed data sets.

Let  $f = (f^1, f^2, \dots, f^m)$  be a nonlinear map from a manifold  $\mathcal{M} \subset \mathbb{R}^n$  to  $\mathbb{R}^m$  such that  $y_i^k = f^k(x_i)$ , where  $f^k$  is at least  $p$ th-order differentiable. For simplicity, and without loss of generality we may assume that  $\mathbf{0} \in \mathcal{M}$  and that  $f(\mathbf{0}) = \mathbf{0}$ . Then the Taylor expansion of  $f^k(x)$  at zero is given by

$$f^k(x) = (\nabla f^k(\mathbf{0}))^T x + \frac{1}{2} x^T H_{f^k}(\mathbf{0}) x + o(\|x\|^2), \quad (31)$$

where  $\nabla f^k$  and  $H_{f^k}$  are the gradient and Hessian of  $f^k$ , respectively. From (31), it can be seen that the linear methods only use the first-order approximation provided by  $\nabla f^k(\mathbf{0})$  to approximate the nonlinear mapping  $f^k(x)$ , while the proposed polynomial mapping contains the extra high-order terms. Therefore, the explicit nonlinear mapping based on the polynomial assumption gives a better approximation to the true nonlinear mapping  $f$  than the explicit linear one.

## 5 EXPERIMENTAL TESTS

In this section, experiments on both synthetic and real world data sets are conducted to illustrate the validity and effectiveness of the proposed NPPE algorithm. In Section 5.1, NPPE is tested on recovering geometric structures of surfaces embedded in  $\mathbb{R}^3$ . In Section 5.2, NPPE is applied to locating new coming data samples in the learned low-dimensional space. In Section 5.3, NPPE is used to extract intrinsic degrees of freedom underlying two image manifolds. In the experiments, the simplified version of NPPE is implemented and compared with NPP [27] and ONPP [30] (which apply the linear and orthogonal linear projection mapping to the training procedure for LLE, respectively) as well as the kernel extrapolation (KE) method proposed in [33].

There are two parameters in the NPPE algorithm, the number  $k$  of nearest neighbors and the polynomial

degree  $p$ .  $k$  is usually set to be 1% of the number of training samples, and the experimental tests show that NPPE is stable around this number. The choice of  $p$  depends on the dimension  $m$ . When  $m$  is small,  $p$  can be large to make NPPE more accurate. When  $m$  is large,  $p$  should be small to make NPPE computationally efficient. Experiments show that NPPE with  $p = 2$  is already accurate enough.

### 5.1 Learning Surfaces in $\mathbb{R}^3$ with NPPE

In the first experiment, NPPE, NPP, ONPP and LLE are applied to the task of unfolding surfaces embedded in  $\mathbb{R}^3$ . The surfaces are the SwissRoll, SwissHole, and Gaussian, all of which are generated by the Matlab Demo available at <http://www.math.umn.edu/~wittman/mani/>. On each manifold, 1000 data samples are randomly generated for training. The number of nearest neighbors is  $k = 10$  and the polynomial degree  $p = 2$ . The experimental results are shown in Fig. 1. In each sub-figure,  $Z = [z_1 z_2 \dots z_N]$  stands for the generating data such that  $x_i = \phi(z_i)$ , where  $\phi$  is the nonlinear mapping that embeds  $Z$  in  $\mathbb{R}^3$ . It can be seen from Fig. 1 that NPPE outperforms all the other three methods, even the LLE method itself. NPP and ONPP fail to unfold these nonlinear manifolds (except for ONPP on Gaussian).

Furthermore, in order to estimate the similarity between the learned low-dimensional representations and the generating data, the residual variance [4]  $\rho(Y, Z) = 1 - R^2(Y, Z)$  is computed, where  $R$  is the standard linear correlation coefficient taken over all entries of  $Y$  and  $Z$ . The lower  $\rho(Y, Z)$  is, the more similar  $Y$  and  $Z$  are. The estimation results are shown in Fig. 1(d). It can be seen that the embedding given by NPPE is the most similar one.

### 5.2 Locating New Data Samples with NPPE

In the second experiment, we apply NPPE, NPP, ONPP and KE to locating new coming samples in the learned low-dimensional space. First, 2000 data samples which evenly distribute on the SwissRoll manifold are generated. Then 1000 samples are randomly selected as the training data to learn the mapping relationship from  $\mathbb{R}^3$  to  $\mathbb{R}^2$  by NPPE, NPP, ONPP and KE. The learned mappings are used to provide the low-dimensional representations for the rest 1000 samples. The time cost of computing the low-dimensional representations of the testing samples is also recorded. Experimental results are shown in Fig. 2. It can be seen that NPPE not only gives the best locating result but also has much lower time cost than KE. NPP and ONPP are faster for computation but fail to give the correct embedding result. The same experiment is also conducted on data samples randomly selected from SwissHole. The results are shown in Fig. 3. NPPE also outperforms the other three methods.

To further validate the performance of NPPE, we randomly generate 11000 samples on the SwissRoll

manifold, 1000 for training and 10000 for testing. The experimental procedure is just the same as the preceding one. Time cost versus number of testing samples is shown in Fig. 4(a). The residual variances between the generating data of the testing samples and their low-dimensional representations given by the four methods, are illustrated in Fig. 4(b). The experimental results show that NPPE is more accurate than all the other three methods with a similar computational cost with NPP and ONPP. Note that, in all the above experiments, the time cost of KE is increasing linearly with the number of testing samples increasing, whilst that of NPP, ONPP and NPPE is almost the same with the increase of the number of testing samples.

### 5.3 Learning Image Manifolds with NPPE

In the last experiment, NPPE is applied to extract intrinsic degrees of freedom underlying two image manifolds, the *l1eface* [2] and *usps-0*.

The *l1eface* consists of 1965 face images of the same person at resolution  $28 \times 20$ , and the two intrinsic degrees of freedom underlying the face images are rotation of the head and facial emotion. We randomly select 1500 samples as the training data and 400 samples as the testing data. The number of nearest neighbors is set to be 15. The experimental results are shown in Fig. 5. The training and testing results are shown on the left and right columns, respectively, in Fig. 5. 100 training samples and 40 testing samples are randomly selected and attached to the learned embedding. It can be seen that NPPE and NPP have successfully recovered the underlying structure of *l1eface*, while the result given by KE is not satisfactory. The rotation degree is not extracted by the learned embedding with KE. Time cost on locating new data samples by these three methods is shown in Fig. 7(a). The time cost of NPPE is higher than that of NPP but lower than that of KE, which supports the analysis of computational complexity in Section 4.2.

The *usps-0* data set consists of 765 images of handwritten digit '0' at resolution  $16 \times 16$ , and the two underlying intrinsic degrees of freedom are the line width and the shape of '0'. 600 samples are randomly selected as training data and 150 samples are chosen to be testing data. The number of nearest neighbors is set to be 5. Fig. 6 illustrates the experimental results. Training and testing results are shown on the left and right columns, respectively. 100 training samples and 20 testing samples are randomly selected and shown in the learned embedding. It can be seen that NPPE has successfully recovered the underlying structure, while it is hard to see the changes of line width and shape in the embedding given by KE and ONPP. Time cost on locating new data samples by these three methods is shown in Fig. 7(b). The time cost of NPPE is higher than ONPP but much lower than KE.

## 6 CONCLUSION

In this paper, an explicit nonlinear mapping for manifold learning is proposed for the first time. Based on the assumption that there is a polynomial mapping from the high-dimensional input samples to their low-dimensional representations, an explicit polynomial mapping is obtained by applying this assumption to a generic model of manifold learning. Furthermore, the NPPE algorithm is a nonlinear dimensionality reduction technique with an explicit nonlinear mapping, which tends to preserve not only the locality but also the nonlinear geometry of the high-dimensional data samples. NPPE can provide convincing embedding results and locate new coming data samples in the reduced low-dimensional space simply and quickly at the same time. Experimental tests on both synthetic and real-world data have validated the effectiveness of the proposed NPPE algorithm.

## REFERENCES

- [1] H.S. Seung and D.D. Lee, "The manifold ways of perception," *Science*, vol. 290, no. 5500, pp. 2268-2269, Dec. 2000.
- [2] S.T. Roweis and L.K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323-2326, Dec. 2000.
- [3] L.K. Saul and S.T. Roweis, "Think globally, fit locally: unsupervised learning of low dimensional manifold," *J. Machine Learning Research*, vol. 4, pp. 119-155, 2003.
- [4] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319-2323, Dec. 2000.
- [5] V. de Silva and J. Tenenbaum, "Global versus local methods in nonlinear dimensionality reduction," *Proc. Advances in Neural Information Processing Systems*, vol. 15, pp. 705-712, 2003.
- [6] I.T. Jolliffe, *Principal Component Analysis*. Springer, 1989.
- [7] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.
- [8] T. Cox and M. Cox, *Multidimensional Scaling*. Chapman and Hall, 1994.
- [9] L. Yang, "Alignment of overlapping locally scaled patches for multidimensional scaling and dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 438-490, Mar. 2008.
- [10] K.Q. Weinberger, L.K. Saul, "Unsupervised learning of image manifolds by semidefinite programming," *Int. J. Comput. Vision*, vol. 70, pp. 77-90, 2006.
- [11] Z. Zhang and H. Zha, "Principal manifolds and nonlinear dimension reduction via local tangent space alignment," *SIAM J. Sci. Comput.*, vol. 26, no. 1, pp. 313-338, 2004.
- [12] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373-1396, Jun. 2003.
- [13] T. Lin, and H. Zha, "Riemannian manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 796-809, May. 2008.
- [14] R.R. Coifman, and S. Lafon, "Diffusion maps," *Appl. Comput. Harmonic Anal.*, vol. 21, pp. 5-30, 2006.
- [15] D. Donoho, and C. Grimes, "Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data," *Proc. Nat. Acad. Sci.*, vol. 100, pp. 5591-5596, 2003.
- [16] J. Lee, and M. Verleysen, *Nonlinear Dimensionality Reduction*, Springer, 2007.
- [17] L. Wang and D. Suter, "Learning and matching of dynamics shape manifolds for human action recognition," *IEEE Trans. Image Process.*, vol. 16, no. 6, pp. 1646-1661, Jun. 2007.
- [18] J. Chen, R. Wang, S. Yan, S. Shan, X. Chen, and W. Gao, "Enhancing human face detection by resampling examples through manifolds," *IEEE Trans. Syst. Man Cybern. Part A*, vol. 37, no. 6, pp. 1017-1028, Nov. 2007.



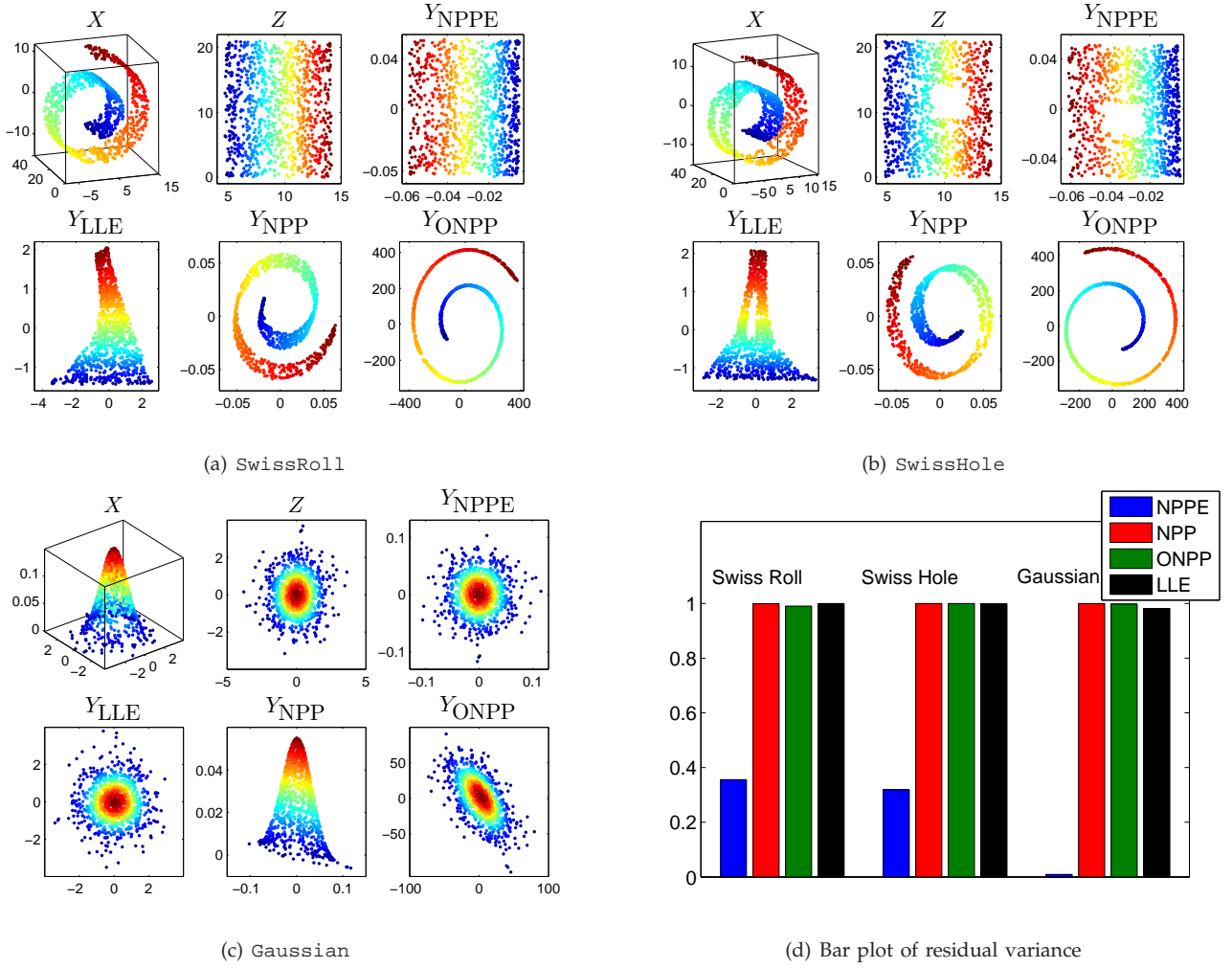


Fig. 1. Experiments on unfolding surfaces embedded in  $\mathbb{R}^3$ . In each sub-figure,  $X$  stands for the training data, and  $Z$  stands for the generating data.  $Y_{NPPE}$ ,  $Y_{LLE}$ ,  $Y_{NPP}$ ,  $Y_{ONPP}$  stand for the embedding given by NPPE, LLE, NPP, and ONPP respectively. (a) Learning results on SwissRoll. (b) Learning results on SwissHole. (c) Learning results on Gaussian. (d) Bar plot of residual variances between  $Y$  and  $Z$ .

- [19] C.M. Bachmann, T.L. Ainsworth, and R.A. Fusina, "Exploiting manifold geometry in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sensing*, vol. 43, no. 3, pp. 441-454, Mar. 2005.
- [20] A. Elgammal and C.S. Lee, "Nonlinear manifold learning for dynamic shape and dynamic appearance," *Comput. Vis. Image Underst.*, vol. 106, no. 1, pp. 31-46, Apr. 2007.
- [21] Q. Wang, G. Xu, and H. Ai, "Learning object intrinsic structure for robust visual tracking," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recog.*, 2003, vol. 2, pp. 227-233.
- [22] H. Qiao, P. Zhang, B. Zhang, and S. Zheng, "Learning an intrinsic variable preserving manifold for dynamic visual tracking," *IEEE Trans. Syst. Man. Cybern. Part B*, in press, 2009.
- [23] H. Qiao, P. Zhang, B. Zhang, and S. Zheng, "Tracking feature extraction based on manifold learning framework," *J. Exp. Theor. Artif. Intell.*, in press, 2009.
- [24] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Advances Neural Inf. Process. Syst.*, 2003.
- [25] X. He, S. Yan, Y. Hu, P. Niyogi, and H.J. Zhang, "Face recognition using Laplacianfaces," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 328-340, Mar. 2005.
- [26] X. He, D. Cai, S. Yan, H.J. Zhang, "Neighborhood preserving embedding," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2005, vol. 2, pp. 1208-1213.
- [27] Y. Pang, L. Zhang, Z. Liu, N. Yu, and H. Li, "Neighborhood preserving projections (NPP): A novel linear dimension reduction method," in *Proc. ICIC (1)*, 2005, pp.117-125.
- [28] D. Cai, X. He, J. Han, and H.J. Zhang, "Orthogonal Laplacianfaces for face recognition," *IEEE Trans. Image Process.*, vol. 15, no. 11, pp. 3608-3614, Nov. 2006.
- [29] E. Kokiopoulou, and Y. Saad, "Orthogonal neighborhood preserving projections," *Proc. Fifth IEEE Int'l Conf. Data Mining*, Nov. 2005.
- [30] E. Kokiopoulou, and Y. Saad, "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, Dec. 2007.
- [31] S. Yan, D. Xu, B.Y. Zhang, H.J. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: a general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40-51, Jan. 2007.
- [32] Y. Bengio, J.F. Paiement, P. Vincent, O. Delalleau, N.L. Roux, and M. Ouimet, "Out-of-sample extensions for LLE, Isomap, MDS, Eigenmaps and spectral clustering," in *Proc. Advances Neural Inf. Process. Syst.*, 2003, vol. 16, pp. 177-184.
- [33] S.V.N. Vishwanathana, K.M. Borgwardt, O. Guttman, and A. Smola, "Kernel extrapolation," *Neurocomputing*, vol. 69, no. 7-9, pp. 721-729, Mar. 2006.
- [34] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: a geometric framework for learning from labelled and unlabelled examples," *J. Mach. Learn. Res.*, vol. 7, pp. 2399-2434, Dec. 2006.
- [35] T. Chin, and D. Suter, "Out-of-sample extrapolation of learned manifolds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 9, Sep. 2008.

- [36] Y. Bengio, O. Delalleau, N. Le Roux, J.-F. Paiement, P. Vincent, and M. Ouimet, "Learning eigenfunctions links spectral embedding and kernel PCA," *Neural Computation*, vol. 16, no. 10, pp. 2197-2219, 2004.
- [37] M. Law, and A. Jain, "Incremental nonlinear dimensionality reduction by manifold learning", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no.3, Mar, 2006.
- [38] C. Baker, *The Numerical Treatment of Intergral Equations*, Clarendon Press, Oxford, 1977.
- [39] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem", *Neural Computaion*, vol. 16, no. 10, pp. 1299-1319, 1998.
- [40] J.R. Magnus, and H. Neudecker, *Matrix Differential Calculus with Applications in Statistics and Econometrics*, Revised Ed., Wiley, 1999.

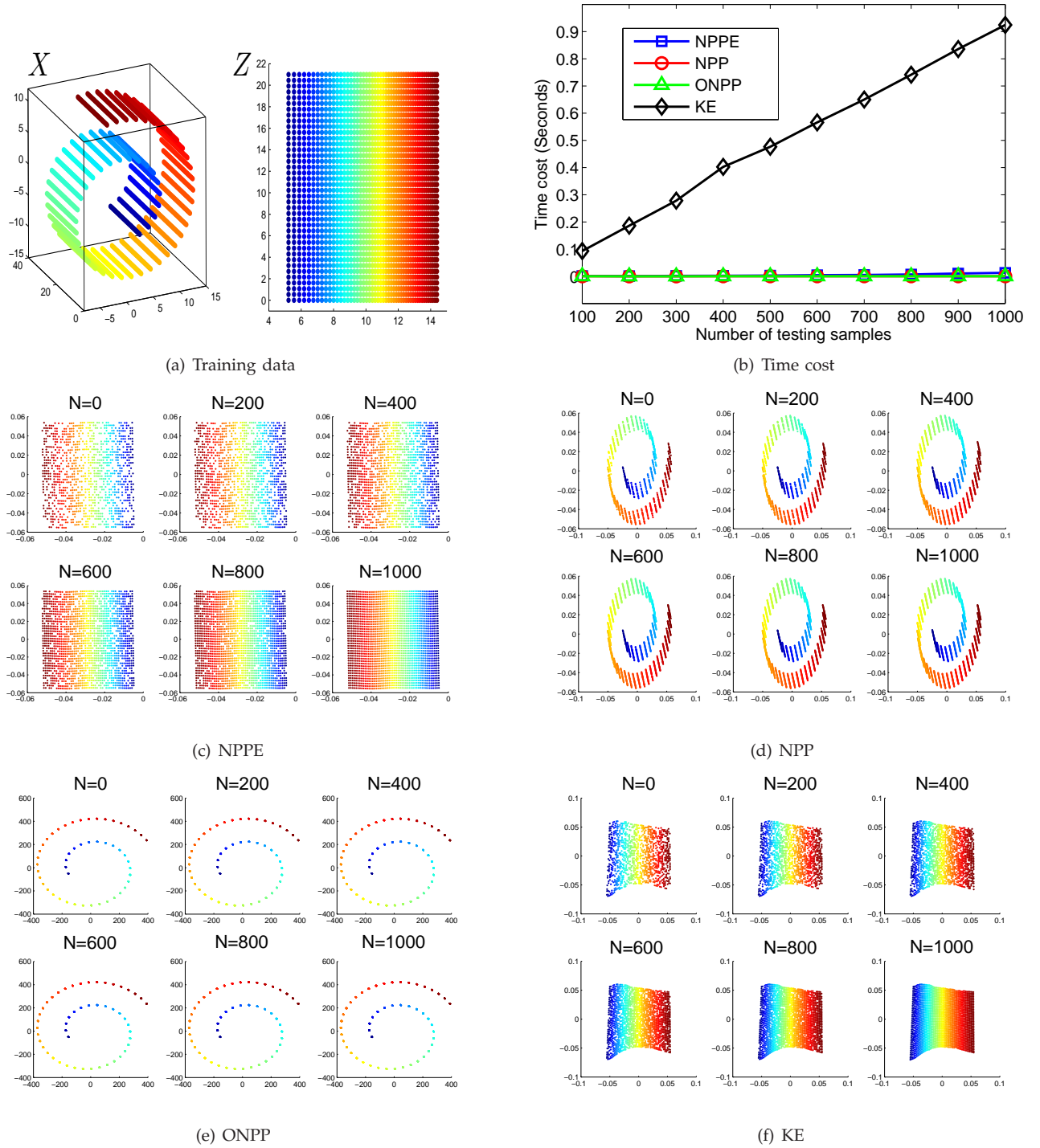


Fig. 2. Experiment on locating new samples for uniformly distributed `SwissRoll` data. (a) Training data and their generating data. (b) Time cost versus number of testing samples. (c) Locating results by NPPE. (d) Locating results by NPP. (e) Locating results by ONPP. (f) Locating results by KE. In (c)-(f),  $N = 0$  stands for the training result.

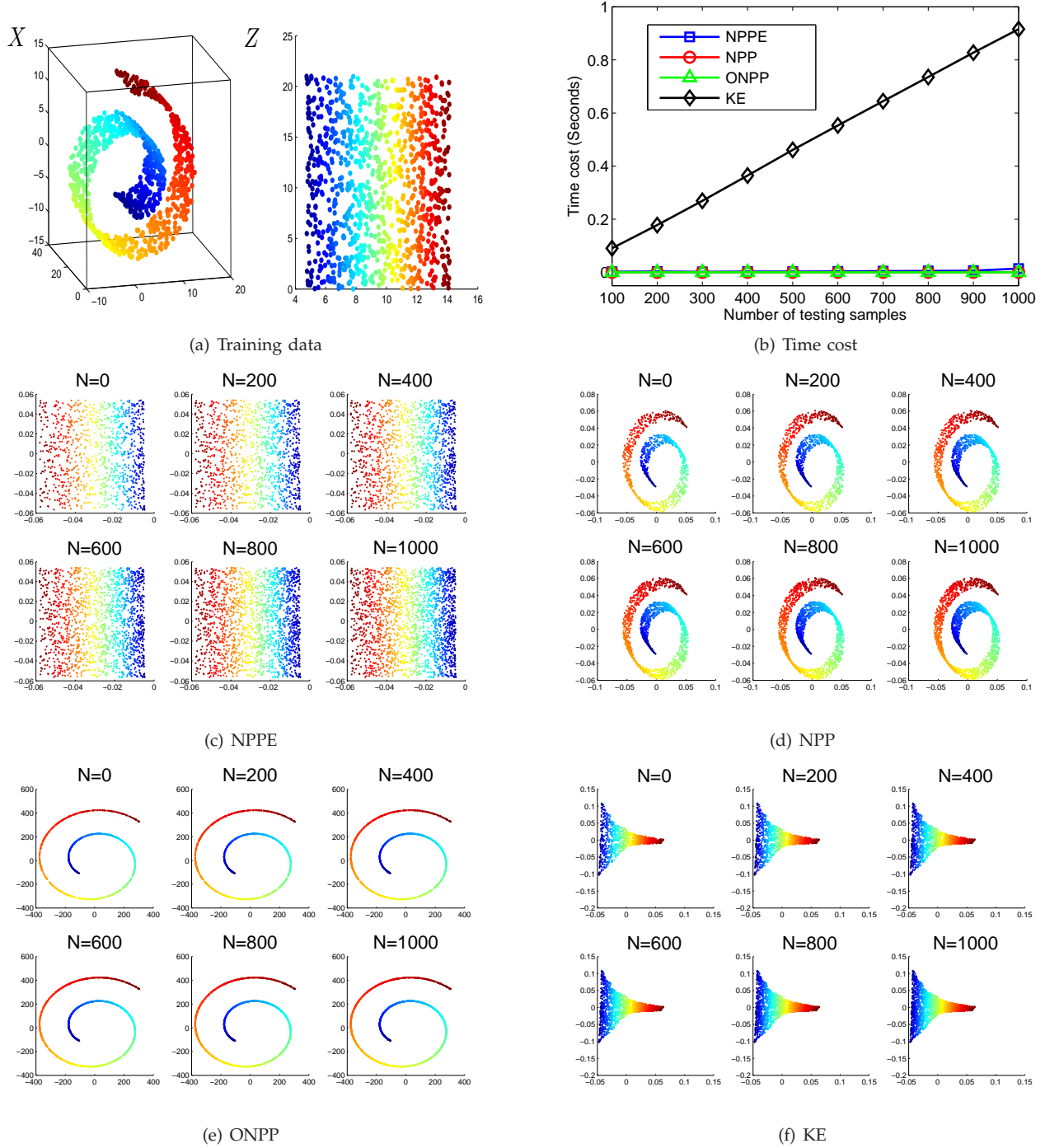
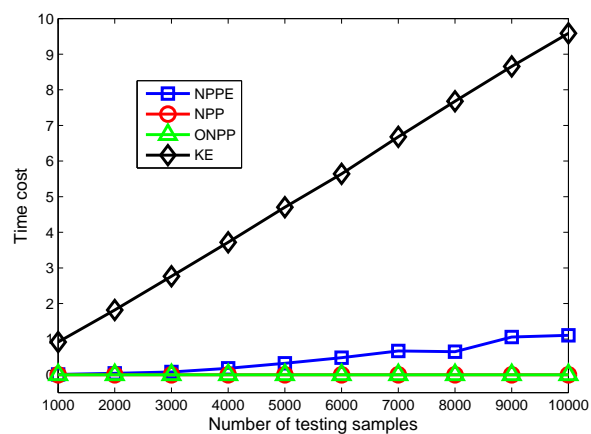
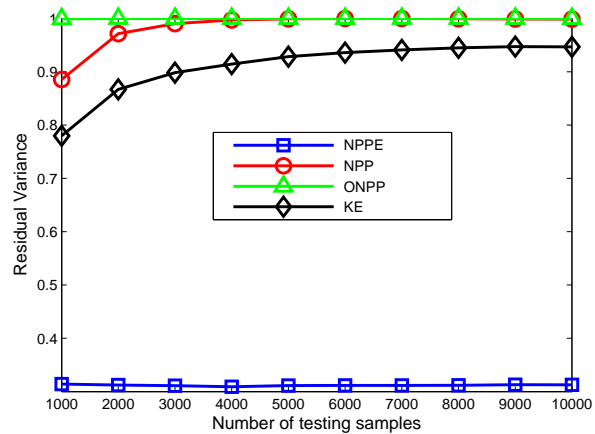


Fig. 3. Experiment on locating new samples for randomly distributed `SwissRoll` data. (a) Training data and their generating data. (b) Time cost versus number of testing samples. (c) Locating results by NPPE. (d) Locating results by NPP. (e) Locating results by ONPP. (f) Locating results by KE. In (c)-(f),  $N = 0$  stands for the training result.



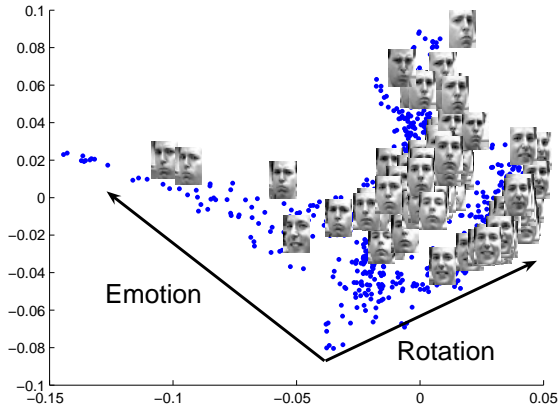


(a) Time cost

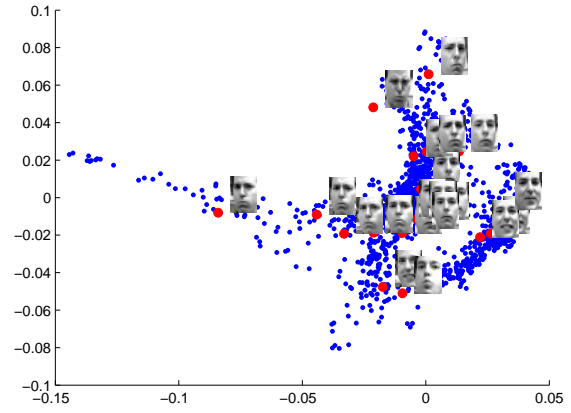


(b) Residual variance

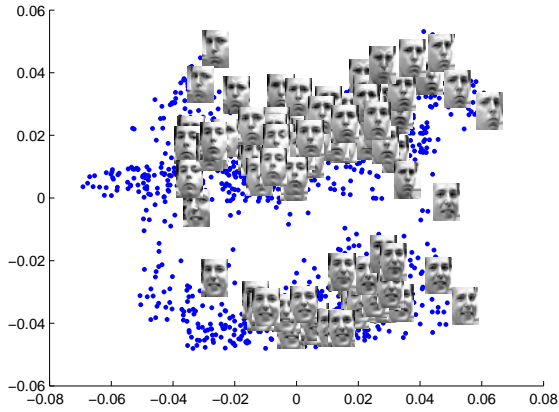
Fig. 4. Experiment on locating new samples for 10000 randomly distributed `SwissRoll` data. (a) Time cost versus number of testing samples. (b) Residual variance versus number of testing samples.



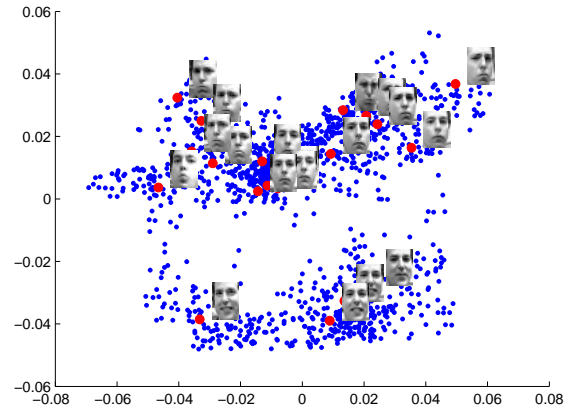
(a) Training by NPPE



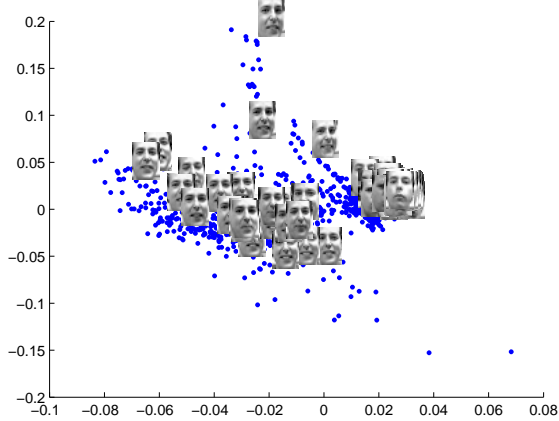
(b) Testing by NPPE



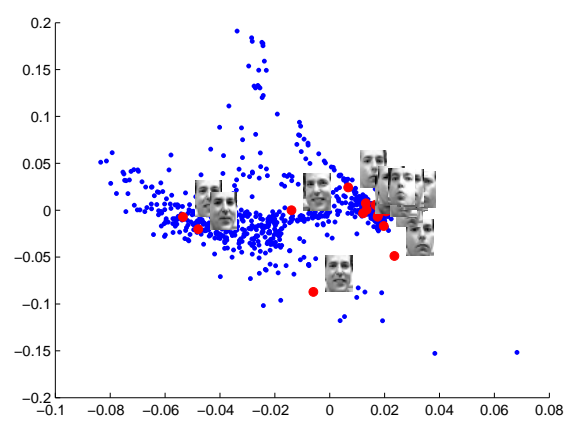
(c) Training by NPP



(d) Testing by NPP

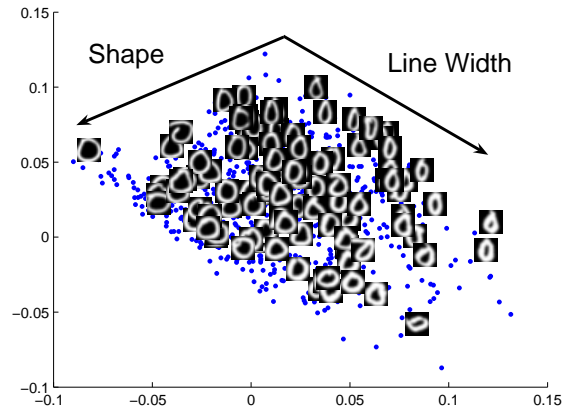


(e) Training by KE

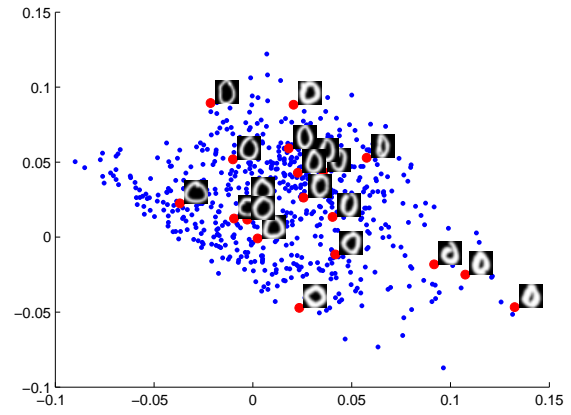


(f) Testing by KE

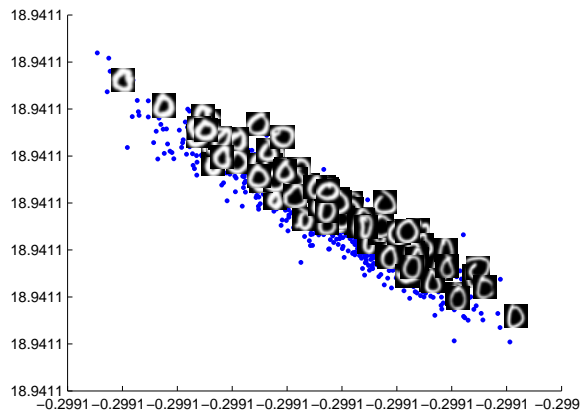
Fig. 5. Experiment on `l1eface` data. Training results are plotted by blue dots while testing results are marked with filled red circles. (a) (b) Learning and testing results by NPPE. (c) (d) Learning and testing results by NPP. (e) (f) Learning and testing results by KE.



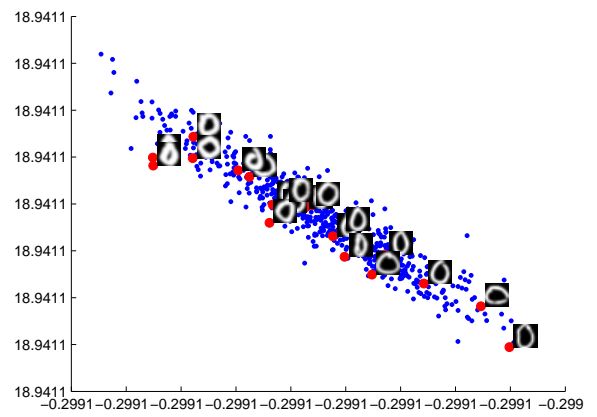
(a) Training by NPPE



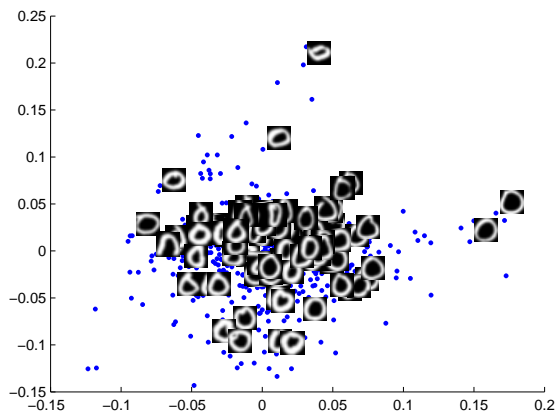
(b) Testing by NPPE



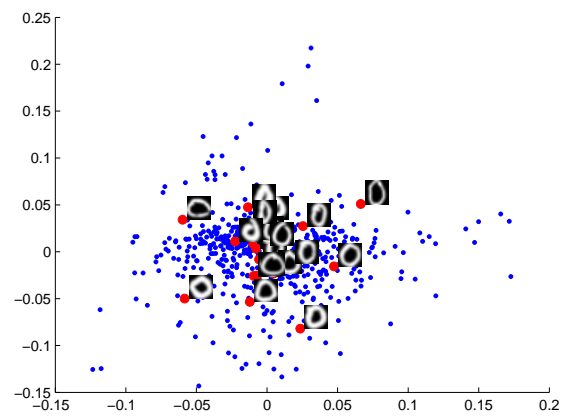
(c) Training by ONPP



(d) Testing by ONPP

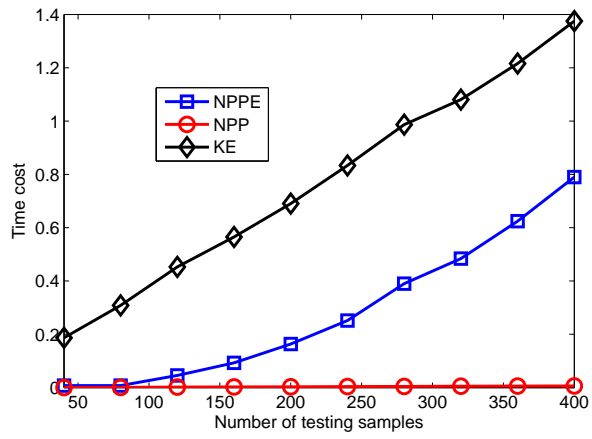


(e) Training by KE

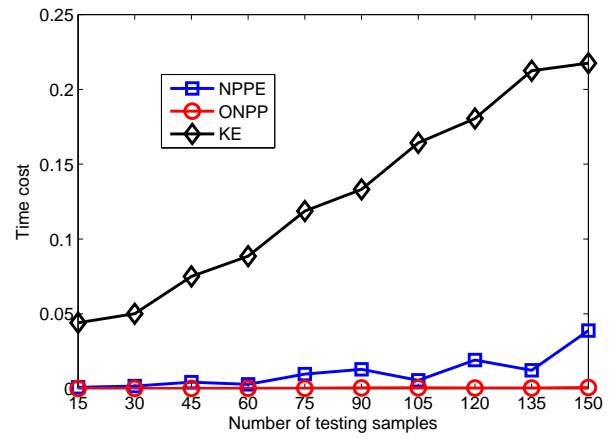


(f) Testing by KE

Fig. 6. Experiment on *usps* data. Training results are plotted by blue dots while testing results are marked with filled red circles. (a) (b) Learning and testing results by NPPE. (c) (d) Learning and testing results by ONPP. (e) (f) Learning and testing results by KE.



(a) Time cost on l1eface



(b) Time cost on usps

Fig. 7. Time cost of experiments on image manifold data. (a) Time cost versus number of testing samples on l1eface. (b) Time cost versus number of testing samples on usps.